Chapter 12
Service Levels, Security, and Trust*

Florian Marienfeld, Edzard Höfig, Michele Bezzi, Matthias Flügge, Jonas Pattberg, Gabriel Serme, Achim D. Brucker, Philip Robinson, Stephen Dawson, and Wolfgang Theilmann

Abstract This chapter covers the scientific background for the Service Level Module of the Unified Service Description Language (USDL). In addition to general service level concepts, we expand on two specific service level fields: security and trust. For that end we first review the state of the art in service level modeling, then we explain the design of the Service Level Module and position it among the rest of USDL. For security, two possible perspectives, a high level business view and a low level engineering approach, are introduced. With regards to trust, USDL is suitable to specify how a service can be rated by its consumers and to ensure that ratings of competing services are comparable, and hence to determine trustworthiness. Additionally, we present a description of non-security-related elements that can be exploited for trust estimation.

* In Alistair Barros, Daniel Oberle (eds.): *Handbook of Service Description: USDL and its Methods*, Part II, Chapter 12, pages 295–326. Springer, New York, 2011.

Florian Marienfeld, Edzard Höfig, Matthias Flügge, Jonas Pattberg
Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany,
e-mail: firstname.lastname@fokus.fraunhofer.de

Michele Bezzi, Gabriel Serme
SAP Research Sophia-Antipolis, 805, Avenue du Dr. Maurice Donat, 06250 Mougins, France,
e-mail: michele.bezzi@sap.com, e-mail: gabriel.serme@sap.com

Achim D. Brucker, Wolfgang Theilmann
SAP Research Karlsruhe, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany,
e-mail: achim.brucker@sap.com, e-mail: wolfgang.theilmann@sap.com

Philip Robinson, Stephen Dawson
SAP Research Belfast, Concourse, Queen's Road, Queen's Island, Titanic Quarter, BT3 9DT Belfast, United Kingdom,
e-mail: philip.robinson@sap.com, e-mail: stephen.dawson@sap.com

12.1 Introduction

The USDL Service Level Module captures concepts concerned with guarantees regarding quality of service operation, which are claimed/requested by different actors involved in the provisioning, delivery and consumption of a service. Given the role of service levels as a vital component of any commercial transaction, it is striking to see how poorly service levels are supported in commercial offerings and also to see the lack of a systematic approach to these in the research arena. Most approaches are missing formal semantics, leave fine-grained content unspecified, lack flexibility and are tuned to specific scenarios and domains. Based on the research results of the SLA@SOI project [21], the USDL effort tries to avoid these shortcomings.

We advocate that a comprehensive, applicable and executable service level metamodel such as the one we contribute is crucial to realize the vision of the Internet of Services (IoS), for it represents a key enabler of effective and efficient service discovery, trade and consumption. It basically gives both, service providers and customers dependability on the quality levels a service comes with and the related obligations of the involved stakeholders.

In our view, two special service level topics deserve to be discussed in more detail in this chapter: security related concepts are actually realized as part of the service level module; trust related elements in contrast, are technically located in other modules, but are conceptually most closely related to SLA. Therefore, they are covered here, too.

There are a number of security focused service description languages, which express the security properties of services and service message exchanges in a standardized way. However, they are often not sufficient to address new scenarios where security is becoming a key aspect in business decisions. The USDL security elements we present in this part of the chapter are a description of security and trust aspects of a service, making the bridge between the IT level and the business level.

As for other USDL modules, the security part provides a minimal set of elements to allow for simple and fast description of security features of services. More technical descriptions can be integrated in the USDL, using appropriate references to standard security description languages.

The question of trust is closely related to security, yet subtly different. When concerned with security, one assumes that chosen business partners, i.e., providers, are both competent and benevolent and considers threats emanating from third parties. With many principals unknown to each other, however, this assumption does not hold, and some prediction has to be made, regarding the reliability of potential partners. The means to perform this calculation and to track and communicate trust information are quite different from those that serve to secure operation against interference from external entities. Nevertheless, trust has a dependency on security, since a smart choice about the provider is worthless if no appropriate security mechanisms are in place. In the traditional world of business services a human user can partly assess the trustworthiness relying on cues like brand of the provider, “word of mouth” recommendations [7] and perceived quality of the website [5]. This does not scale to the level of an Internet of Services, where services are automatically com-

posed and delivered with limited human intervention, and explicit trust and security properties are becoming a key for a broad adoption of service technology [17].

The structure of this chapter is the following: In the subsequent Section 12.2 we review related approaches. In Section 12.3 we position the Service Level Module among the other modules and layout its content. Section 12.4 introduces the USDL representation of security properties. Trust awareness is covered in Section 12.5. Conclusions are drawn in the Section 12.6.

12.2 State of the Art

The state of the art in SLA specification and management motivates the features of USDL. USDL is a synthesis and generalization of existing specifications, capturing the essential elements of an SLA specification. In doing so it satisfies key requirements for SLA management and enables new business-oriented aspects of service management (e.g., security and privacy) to be encoded in SLA documents. SLA Management includes the specification of machine and human readable documents, the configuration of systems based on the content of these documents and the monitoring of parameters expressed in these documents in order to achieve compliance. Failure to achieve SLA compliance can lead to losses in efficiency, performance, reputation and business opportunities. This section discusses the state of the art in SLA specification, monitoring, negotiation and enforcement as well as security models. A comprehensive solution to SLA management addresses each of these areas. For each area of SLA management the requirements and challenges are described, such that existing approaches can be compared to the SLA principles of the USDL Service Level Module.

12.2.1 SLA Specification

SLA specification can be viewed as both a process and a document. It is the process of a service consumer initially declaring and agreeing to specific service requirements, when entering a contract with a service operator or provider. The consumer hence specifies *Service Level Objectives (SLOs)*, before or after the availability and capability of services and service providers are known. In the case this is done beforehand, SLOs are used for service discovery. If SLOs are specified when the service and provider are known, this is the process of initiating or requesting the provisioning of a specific service on their behalf. From the provider's perspective it is the process of declaring their service capabilities and quality guarantees in a form of advertisement. This is also known as a *Service Level Agreement Template (SLAT)*, and acts as the baseline for contractual agreement with customers, potentially in different classes. The specified documents all have the same minimal requirements for structure and content:

- Definition of individuals, organizations and roles involved in the agreement. The roles are typically consumer/customer and provider/operator, but can also include a third-party broker, an intermediate actor in the SLA management process.
- Functional description of the service's purpose and capabilities. In the case of a technical, IT service, such as a Web service, the functional description refers to the set of operations, methods and parameters. For example, the Web-Services Description Language (WSDL) provides a standard specification for SOAP-based Web services.
- Costs to the consumer for receiving the service. The units for costs are defined by relating financial costs to utility functions of the resources consumed by the service. For example costs can be defined per requests, per volume of storage used, per user or on a fixed-term or unlimited basis.
- Guarantee or Quality of Service (QoS) terms define the non-functional properties of the service. These properties include availability, performance, response time, reliability and security.
- Compensation terms define what the consumer can rightfully demand from the service provider in return, should the functional or guarantee terms not be fulfilled.

The distinction between a SLA document and a configuration document for a service infrastructure is becoming increasingly fuzzy. The contents of a SLA are inevitably translated into concrete configuration directives that are used to guide the provisioning of resources, deployment of software and tuning of settings to enable effective operation of the service. Effective operation means that all functional and non-functional terms in the SLA can be satisfied without exploding costs for the provider. Providers need to keep their costs down so that they can offer an attractive service deal to consumers without sacrificing their profit targets. Given that this becomes a more complex problem for human operators to deal with manually and rapidly, the following new requirements arise for SLA specifications in a world of service management automation:

1. SLA specifications need to be both human and machine readable, given that people need to define, exchange and agree terms, while algorithms are required to parse, extract and analyze terms, and translate terms and analysis results to configuration directives in an efficient and accurate manner.
2. Traceability to organizational Key Performance Indicators (KPIs) such as performance, availability and cost.
3. All terms must be associated with concrete metrics that enable them to be monitored and audited.
4. Sufficiently flexible for application in different service operation domains and contexts.
5. Support for the entire SLA life cycle, including negotiation, provisioning, monitoring and decommissioning.

There are various specification languages in existence, each with different motives but similar concepts. The Open Grid Forums's WebService-Agreement [1] and

IBM's Web Service Level Agreement (WSLA [12]) are the most well-known languages for expressing SLAs. WSLA is a comprehensive specification for describing SLAs for Web services, providing a template for defining concrete metrics. WSLA however does not support the entire life cycle of SLAs, as negotiation of terms is not supported for flexibility. WS-Agreement superseded WSLA in order to address these points. While WS-Agreement is very flexible, given its generality, and provides a comprehensive schema that enables human and machine interpretation, it still requires more concrete metrics and explicit traceability to KPIs. Another point to mention is that WSLA and WS-Agreement were developed for WSDL-type Web services and hence do not have the semantics included for dealing with non-WSDL services.

There are other specifications existing that are not tied to WSDL. SLAng [9] for example is specified in the Object Management Group's (OMG) Meta Object Facility (MOF) and, thus, has a degree of language independence with mappings to XML and Human-Usable Textual Notation (HUTN). SLAng also places greater emphasis on semantics, providing formal notions of SLA compatibility, monitorability and constrained service behaviour. It is, however, targeted at electronic services and provides only a limited set of domain-specific QoS constraints. Another language, viz., CC-Pi [2] is more generic, offering a theoretical framework for mapping SLAs to service constraints. The CC-Pi model is, however, tightly-coupled to the mechanics of negotiation, and does not address common constructs such as agreement party details or service interfaces. The SLA* approach from SLA@SOI [8], in contrast, is a complete abstract SLA syntax which has been designed to be independent of underlying technologies. It is decoupled both from particular notions of service, and from particular modes of expression, and can be extended to diverse scenarios without sacrificing formality or semantics.

12.2.2 SLA Monitoring

The formal specification of SLAs enables monitoring the status and compliance of services with the organizational KPIs of the parties involved. As stated in the previous section, in order to monitor aspects of a service's operation, it must be possible to measure that aspect using concrete metrics. There are critical areas of service and resource management that depend on this information and delivering it to the right people and systems in a timely manner. These are discussed in the following and are the set of standard activities defined for IT Service Management (ITSM) in the Information Technology Infrastructure Library (ITIL).[3]

Capacity planning and management involves the allocation of resources in order to avoid over-spending, wastage and under-provisioning. However, there is still the over-arching objective of satisfying the obligations and guarantees stated in SLAs. Resources include people, hardware, software licenses, software in-

[3] `http://www.itil-officialsite.com`

stances and materials. SLA monitoring is the timely update of information about resource capabilities, availabilities and performance during operation. In order for services to be delivered efficiently in an on-demand manner, there is a need for mixing historical, predictive and live information about resources for dynamic replanning and provisioning of resources. In the case of people this includes the assignment of tasks and access to resources through justifiable provisioning of users and assignment of privileges.

Availability analysis and management is the set of activities done to maximize the likelihood that resources will be available when required, as well as recoverable from unsafe or fault states. Availability is both an explicit and implicit objective of SLA management. As an explicit objective the guarantees of availability and recovery are agreed in the SLA. As an implicit objective an analyst or management system needs to monitor the resource to identify when associated services are required. This also has relations to capacity planning and management, as the sizing and number of resources will change the availability of services.

Operations management is the coordination tasks and processes amongst resources to ensure that the KPIs of an organization are met, along with the objectives in the SLAs the organization has with customers and partners. SLA monitoring should provide feedback about the current load on different resources and the criticality of service request to be handled. Operations management usually includes optimization of handling service requests and assigning tasks based on multiple objectives. These multiple objectives are derived from the set of objectives in multiple SLAs, such that conflicts and contentions will arise in an environment that allows concurrent services and service users.

Incident management is the handling of inevitable failures and unexpected events that arise during service operation. Capacity planning has to take incident management into account, as redundant resources and back-up resources usually need to be deployed to for executing contingency plans. Incident management is also related to availability analysis and management, as effective incident management increases the likelihood that resources and services will be available even if experiencing known and unexpected incidents. Operations management also extends to incident management, as the coordination of resources during contingency and recovery operations might be more critical than during normal operation. SLA monitoring is hence critical for identifying and characterizing incidents and deviations from SLA obligations and guarantees, such that effective incident response actions can be executed.

SLA monitoring is more than blanket monitoring of every possible operation, property and behavior of resources, although this might be necessary to some extent. SLA monitoring has to be purposeful and driven by measurable indicators derived from business objectives. Even if there is extensive monitoring of all resources, a system of filtering and routing information to meaningful endpoints or sinks is necessary. Failure to meet this requirements results in monitoring information consumers being overwhelmed, the processing of irrelevant or redundant data, and the introduction of unnecessary communications and processing bottlenecks.

Figure 12.1 shows a conceptual architecture for SLA monitoring for the purpose of discussing the state of the art as opposed to proposing a blueprint.

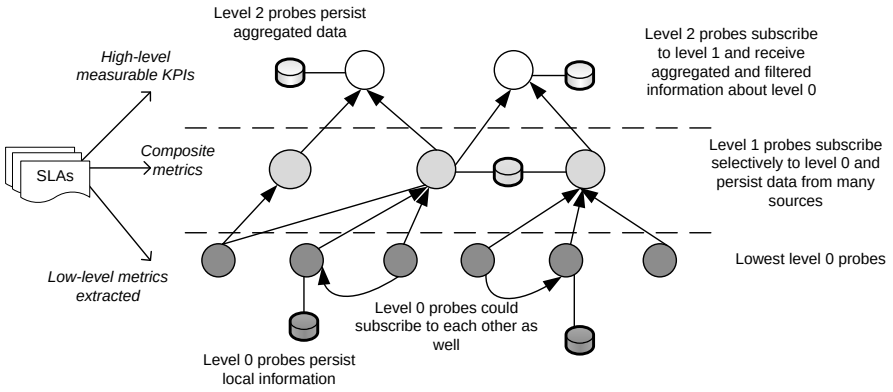

Fig. 12.1: Conceptual architecture for SLA monitoring.

The typical components in a SLA monitoring solution are collections of probes or sensors that gather localized resource information and publish it to a set of information subscribers. The assignment of subscribers to channels that the probes publish on is then based on the types of services, location of resources, connectivity and the respective metrics associated with services. This assures that the collection and distribution of monitoring data is traceable to specific KPIs and operational contexts. Higher-level probes subscribe to lower-level probes and hence act as data aggregators, filters and transformers, based on the information needs of the SLA management system at the time of deployment. The needs for monitoring are determined by analyzing the existing SLAs. Very low-level metrics such as CPU, memory and network utilization and availability can be mapped directly to localized probes that gather raw resource performance information. Higher-level metrics that described collections of resources, functions (e.g., predictions and trends) require higher-level probes to subscribe to lower level probes. These are referred to as level 1 probes in Figure 12.1. The information required to configure monitors is extracted from SLA specifications.

Monitored data can be persisted at each level in order to have different levels of granularity for historical data. Determining when to capture, aggregate, archive and delete data will differ from solution to solution. Furthermore, the selection of where to persist data is dependent on the local storage capabilities of the respective monitored resources, as well as the overall storage architecture and network topology. Centralized persistence has the advantage of simplicity and faster queries, as data is stored in one location. The disadvantage is the single point of failure that can cause all historical data to be damaged or lost. The loss of historical data can be problematic for optimizing the way in which SLAs are enforced and the availability of audit data when payment is due or disputes arise.

12.2.3 SLA Negotiation and Enforcement

SLA negotiation is the process of a service provider and consumer reaching consensus on the terms to be included in a SLA document. Negotiation is complete when all parties agree to the terms. This process can be automated but is often done as manual exchanges of proposals/tenders and offers. A generic protocol for negotiation is known as the *alternating offers-based protocol* [24], which is shown in Figure 12.2.

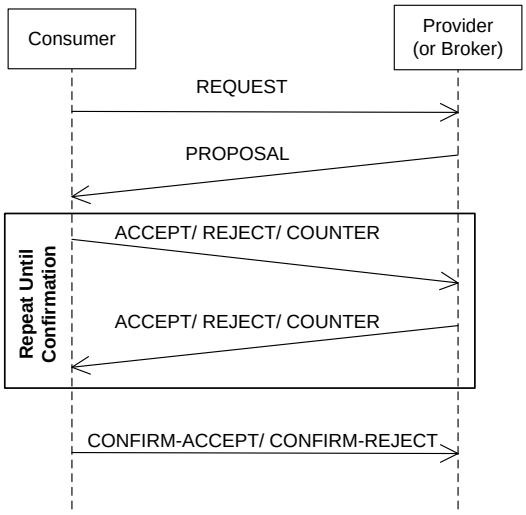

Fig. 12.2: Alternating offers-based protocol adapted from Venugopal, Chu and Buyya [24].

The protocol begins with the consumer (or initiator) sending a request to the provider or broker, who take the role of responder. The provider issues a proposal of how they can satisfy the request. For example, the request might have stated "*provide service X with a guarantee of less than 3 ms response time for 1000 concurrent users*." The proposal would use the same functional specification and quality metrics although the provider might not be capable of exactly matching the request. The provider might offer a proposal such as "*Can provide service X with a guarantee of less than 3 ms response time if there are less than 750 concurrent users*." The consumer can accept, reject or counter (i.e., update the request) the proposal, to which the provider can do the same until a confirmed state is reached. The confirmed state can be either of acceptance or rejection. This plain protocol assumes that both parties will adhere to the protocol and that the provider will typically have counter offers. However, the style of negotiation varies based on the number of parties involved and the set of options available. Three styles of SLA negotiation are as follows:

- **Boolean**: there are no alternatives offered by providers. Consumers either accept or reject a provider's offer without requesting alternatives. This style of negotiation is typically brokered, as the consumer considers the offerings of multiple providers registered with the same broker. Consumers select providers that have offers that best fit their requirements. The proposal is that of a single provider and counters are alternative providers. It can be the case that the provider chooses to maintain a very generic, one-size-fits-all policy for services to avoid any liability. However such a style of negotiation is not attractive for critical services where the consumers need to know what to explicitly expect from the provider.
- **Template**: in this case a provider has several options in the form of templates. Examples of this are the Amazon EC2 services[4] that offer T-shirt sizes of small, medium, large and x-large, which all have predefined service guarantees and technical specifications. This style has issues for over and under-provisioning, as the capabilities of the provider might change over time. They might need to dynamically update the specifications of their templates and offerings based on their current and planned prediction.
- **Scalar**: the most complex style of negotiation is where fine-grained adjustments are permitted on a dynamic basis. The provider does not counter with a suite of templates but makes adjustments in their guarantees and obligations, which then have an impact on how resources are sized and configured. Elastichosts[5] are a provider of infrastructure services similar to Amazon, but allow customers to state the exact amount of capacity (memory, storage and CPU) they require.

Given that the style of negotiation varies from business domain to business domain, the Service Level Module must be sufficiently configurable to support any of these styles. Enforcement of SLAs is the translation of the terms in the agreement to concrete configuration directives. There are three possibilities that exist for handling the translation of SLAs to directives, each having their advantages and disadvantages for building a complete solution.

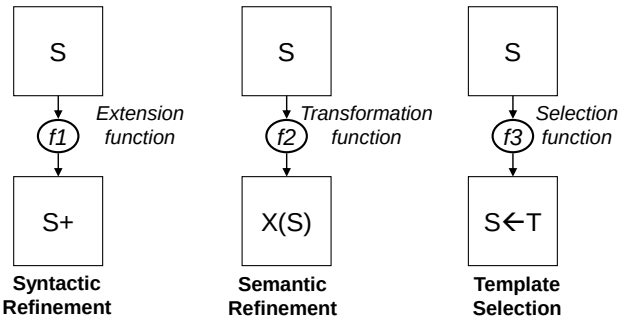

Fig. 12.3: Different approaches to SLA translation to configuration directives

[4] http://aws.amazon.com/ec2

[5] http://www.elastichosts.com

The advantages and disadvantages of these different approaches are discussed below, as it is important for any user of the Service Level Module to know which approach should be used and the consequences of that selection.

1. **Syntactic Refinement**: In this case there are homogeneous modeling semantics for SLA terms and configuration directives. It is only a case of adding more details (e.g., missing parameter values) to the specification without changing the semantics or schema of the specification. Syntactic refinement is hence transforming a SLA model S to a configuration $S+$. For example, S could be WSLA or WS-Agreement specification with many null fields, where every $S+$ is more information provided to those fields.
   - *Advantages*: simpler process for moving through the SLA transformation process as there are less processing and transformation logic involved. This also implies higher scalability, easier rollback, consistency checks and simulation.
   - *Disadvantages*: there is the initial agreement that all management components use the same semantics. Secondly, some of the human readability would have to be sacrificed in order to have a specification that is already at the level of configuration directives without the ability to separate concerns.
2. **Transformation (Semantic Refinement)**: In this case there is no assumption of homogeneous specification templates, such that the SLAs can be specified in any language. In order to obtain configuration directives, the specification would then have to be transformed into lower level semantics using a transformation function. The transformation function must be provably correct for deriving a specification with different syntactical properties. There is also need to add annotations to the initial specification in order to supply sufficient information for concrete configuration directives. In Figure 12.3 this is illustrated as performing a transformation function $X(.)$ on the specification S, such that $X(S)$ is produced, which could be a totally different format for specifying configuration directives.
   - *Advantages*: lower coupling and dependencies between components that handle SLA specifications and configuration directives. Existing control and management components do not have to be changed in order to configure resources based on SLA specifications. There is also better support for human and machine readability, as S could be intended for humans and $X(S)$ is compiled for machines.
   - *Disadvantages*: transformation can be quite complex and hence could introduce errors that take a long time to debug; additional model annotations would have to be introduced in order to perform this automatically.
3. **Template Selection**: In this case there are no assumptions of homogeneous models or templates, but there is a logic implemented for mapping higher level models to lower level deployable component templates, in such a way

that selected templates are understood as being conformant with a higher-level model specification. Figure 12.3 illustrates this by showing the template selection $S \leftarrow T$, where the template T is selected given the specification S. The set of templates are discrete and predefined.

- *Advantages*: even lower coupling and dependencies between models; selection logic is relatively easy to encode and tolerates manual interaction. Higher flexibility gained in how the SLA is specified. Templates can be made to be platform independent, such that configuration directives can be further compiled for different management systems.
- *Disadvantages*: carries the additional overhead of creating templates and loses the dynamic property of the above methodologies.

This coverage of the state of the art in SLA management shows that there is still a gap in the areas of SLA specifications that are flexible for multiple domains and not restricted to IT-centric services. Secondly, there should not be an assumption of what level of monitoring is going to be associated with the SLAs specified in the language. Finally, the language needs to be sufficiently flexible and expressive to support different forms of translation into configuration directives, without sacrificing human-readability and machine processing that enable advanced analytics and automation in service management.

12.3 The Service Level Module

12.3.1 Position within USDL

The Service Level Module is an integral part of USDL. The module ties together all the functional and non-functional guarantees that are stated on top of a core service description (as described in the Service Module, cf. Chapter 13). Furthermore, all guarantees are clearly linked to its related (and probably obligated) stakeholder (as expressed in the Participants Module, cf. Chapter 13). Last there is a strong semantic linkage to the Legal Module (cf. Chapter 10). While the Legal Module describes the constraints and aspects of licensing, the Service Level Module complements this with the specific conditions that are to be guaranteed.

The Service Level Module directly responds to the main requirements for USDL. It satisfies a clear *Conceptualization* (see Section 8.3.1.1) as it realizes the core principles of guarantee, obligated party, affected service elements and negotiable parameters on an abstract level. It comes with means for *Extensibility* (see Section 8.3.1.4) that allows for incorporating arbitrary, domain specific type/term systems. It also supports *Comprehensibility* (see Section 8.3.1.5) as it allows description of service levels in human-readable, semi-structured, and fully structured ways. *Organizational Embedding* (see Section 8.3.2.1) is achieved by the association to participants. And last, it supports *Deployment Symmetry* (see Section 8.3.2.4) as

it allows mutual, symmetric obligation relationship between arbitrary roles in the service value creation chain.

12.3.2 Construction Rationale

Service Level Agreements, as considered in the research community, specify all the conditions under which services are to be delivered. In that sense the whole specification of USDL can be considered as a language to describe SLAs.

However, within USDL several key perspectives have got high priority and likewise shall get high visibility. For this reason the actual Service Level Module is just one module among others (such as pricing or legal).

The Service Level Module is intentionally kept completely generic, as it does not specify how concrete service levels on concrete aspects (such as legal, pricing or security) shall be specified. Instead, its main purpose is twofold. First, it shall provide a proper glue between other USDL concepts. For example it specifies to which elements of a function a certain service level shall apply and who is the related stakeholder. Second, it should allow for incorporation of arbitrary attribute and expression languages. One particular attribute language is specified further below in this chapter and deals with security aspects. Other languages could be integrated as well, e.g., the UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE) or the full SLA model from SLA@SOI.

The Service Level Module intentionally does not specify any concrete attribute type systems. This is done for three reasons. First, there is no commonly agreeable type system that applies to all kinds of domains. Secondly, even some common core for such a type system can easily grow to extensive size and therefore contradicts our ambition to keep the Service Level Module as lean as possible. Third, there are already established type systems for different domains, and we wanted to keep the Service Level Module neutral and fully extensible towards these type systems. However, for pragmatic reasons, the Service Level Module also comes with a base extension (not part of the core USDL language) which provides common notions for frequently used metrics such as reliability, security, location, time, performance, and availability.

12.3.3 Module Overview

The main concepts of the module (see Figure 12.4) are the ServiceLevel (specifying either a state or an action), the ServiceLevelExpression, and the ServiceLevelAttributes. Furthermore, the module contains important references to other modules' concepts such as the ObligatedParty (Participant Module), the VariableDeclaration (Foundation Module), and the relatesTo reference (Foundation Module).

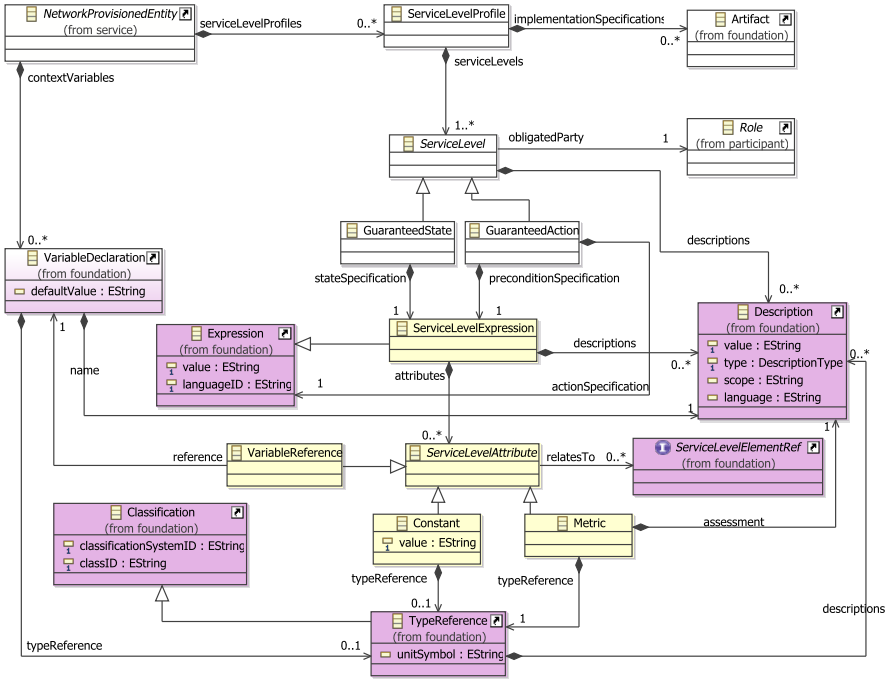

Fig. 12.4: Overview of the Service Level Module.

The entry point to specify the service levels of a given service is realized via ServiceLevelProfile. A set of service level specifications are combined into one profile and are offered, negotiated, or agreed upon as a whole. Different profiles can be used to specify different options of how service levels may be specified and grouped (e.g., as gold, silver, bronze profile). A ServiceLevelProfile resembles the concept of a Service Level Agreement Template as for example specified in WS-Agreement.

A ServiceLevel specifies a single service level objective as it characterizes an offered, negotiated or agreed service. ServiceLevels are defined by the parties participating in service provisioning, delivery, and consumption and express assertions that are claimed or expected to hold during these activities. Such assertions are always attributed to a single party, which is obligated to enforce the service level. From the viewpoint of the party defining the ServiceLevel two cases are distinguished. Either the defining party obligates itself to ensure the ServiceLevel, i.e., it claims that the assertion will hold, or the defining party expects the obligated party to ensure the ServiceLevel, i.e., it requires the other party to enforce the assertion. A ServiceLevel can be either a GuaranteedState (specifies a single state that must be maintained within the lifetime of any service instance, to which the respective service level profile applies) or a GuaranteedAction (specifies a self-contained activity that must be performed, if and only if during the lifetime of any service in-

stance to which the respective service level profile applies a specific precondition is fulfilled).

A ServiceLevelExpression specifies an expression that is evaluated in the context of a service level state or action. For this purpose it may reference a set of ServiceLevelAttributes (constants, metrics or variable references) and define relationships between these attributes, e.g., Boolean or arithmetic operands. Typically, it resolves to a Boolean value that indicates whether a GuaranteedState is met or whether the precondition to a GuaranteedAction is fulfilled.

A ServiceLevelAttribute specifies a single attribute that is part of a service level expression. Attributes can take various concrete forms, of which three (constant, variable reference and metric) are defined in the core version of USDL. A ServiceLevelAttribute has a scope, i.e., it exists in reference to something to which it applies. By default, all attributes are defined in relation to the entire service (including its overall context). Alternatively, a scope that covers only parts may be specified. A Constant specifies a single ServiceLevelAttribute which is constant during service operation, i.e., during the lifetime of any service instance. A Metric specifies a single ServiceLevelAttribute which refers to the observation (measure) of a property of the service at service runtime. It may change over the lifetime of a service instance. Last, a VariableReference allows for referencing a variable declared in the global context of a service or service bundle. VariableReferences are used, for instance, as part of service level expressions.

In Listing 12.1 (cf. appendix of this chapter) we provide a more detailed example of how the Service Level Module can be used. It is related to a 3PL logistics provider as specified in the running example introduced in Chapter 8. The XML snippet shows a service level specification, where

1. Customers can specify their expected delivery duration
2. A provider-obligated term guarantees the delivery within the specified duration (ServiceLevel nb 1)
3. A provider-obligated term guarantees that goods are maintained at -5 degrees Celsius (ServiceLevel nb 2)
4. A provider-obligated action regulates penalty payments for delayed deliveries (ServiceLevel nb 3)

12.4 USDL and Security

The implementation of security measures for electronic services, such as the proper authentication of consumers and the encryption of service data, can become a complex and error-prone effort. Service providers do not always possess sufficient knowledge and experience on technical security mechanisms and standards. The same applies to the definition of service security characteristics as part of a formal service description. Such technical issues may be far out of the core competencies on which a service provider should be able to concentrate in a globalized and competitive market.

With regard to the security issues mentioned above, this means that platforms on which business services are developed, hosted and traded will provide — as part of the platform infrastructure — shared security services to handle the management of identities, the access control to services, data protection and related tasks. Business services strongly vary in their security requirements. General information services, such as rental car availability information, and transaction oriented services, such as the booking of a rental car in combination with a debit advice, are possibly traded on one and the same platform. Hence, security services provided by the platform are not statically bound to business services. Rather they can be integrated with and bound to business services on demand and in a flexible manner (Security-as-a-Service).

Ideally, a business service provider is enabled to implement service security measures in a declarative manner, i.e., by specifying the desired security characteristics as part of a service model or service description. In this case the underlying platform (operated by a third party) takes care of fulfilling the declared security requirements by including the required security services, e.g., by enforcing user authentication or by ensuring non-repudiation of the service usage. In order to anticipate the potentially limited technical security knowledge of business service providers, the description of security characteristics should be supported on an abstract, non-technical level. The abstract description of business service security characteristics also enables non-technicians on the consumer side to express their security demands and to find business services that comply with these demands.

The USDL (v3 Milestone 5 specification) elements that serve to model these claims and requirements are depicted in Figure 12.5. The key building blocks are the classes SecurityMetric and SecurityAttribute, which are explained in detail here. They are special cases of the class ServiceLevelAttribute and both entail properties of the enumeration types SecurityGoal, SecurityRequirementLevel and RealizationLevel. The former two are straightforward elements that can be used as detailed below. RealizationLevel refers to the OSI reference model [3] for layered networks and specifies at what layer in the communication stack the security element is targeted at. These classes, that were introduced due to security considerations were placed in the module Service Level Base Extension. On the one hand, non-functional security properties are clearly part of the agreed service level. On the other hand, they are considered a domain specific extension and not part of USDL since they are not inherently part of all domains of business services, but only geared towards domains involving extensive network communication.

The SecurityAttribute and SecurityMetric elements support the description of service security characteristics in two alternative, mutually exclusive ways, they mainly differ in the abstraction level. SecurityAttribute specifies security characteristics in terms of security goals and security requirement levels, but it does not provide reference to specific technical security measures. Such SecurityGoals are integrity, confidentiality, identification, authentication, authorization, privacy and accountability. A SecurityRequirementLevel is used to indicate the desired implementation degree of a security goal, i.e., the required level of protection/security with respect to the security goal. The granularity of the security requirement levels is inspired by the "authentication assurance levels" [23] as developed by the Euro-

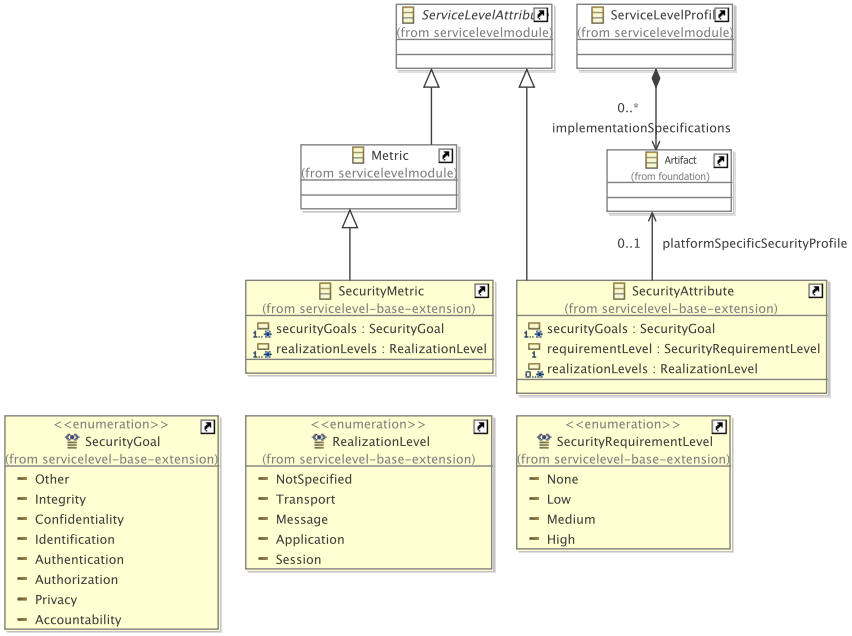

Fig. 12.5: Model elements introduced for security infrastructure.

pean STORK[6] project. On the other hand, SecurityMetric describes the high level security goal, but also the specific security mechanism used to address this goal, possibly including values of parameters and/or pointers to concrete security policies written in a standard policy language, such as WS-Security, P3P, XACML.

USDL allows one to browse services, and select them according to their capabilities and features, and security features may be important criteria for such a choice. Thus, USDL elements for security can express claims and/or requirements about security properties, with information on protections that are enforced by the service provider. For example, a service provider may claim that the customer data remain confidential, but at the same time it may require that the consumer should support message encryption to send input data in a confidential way.

In the next subsections we will describe the two approaches in more detail.

12.4.1 SecurityAttribute

Defining security characteristics in terms of security goals and requirement levels enables actors (on the provider as well as on the consumer side) who are not "techni-

[6] `https://www.eid-stork.eu`

cal security experts," i.e., who are not familiar with, e.g., WS-Security policies [11] or XACML [16] statements, to express their security demands. These requirements are then interpreted in the context of the particular platform that provides access to the service, using a platform-specific security profile (additional service metadata). The platform may undertake the task of implementing the security goals at the desired requirement levels by generating appropriate technical security policies (e.g., WS-SecurityPolicy artifacts) and by involving suitable platform services that are handling authorization, authentication, encryption etc.

The platform operator may create a "platform-specific security profile" to specify the technical details on how a security goal is realized on a given platform. The platform-specific security profile maps security goals that are defined at certain security requirement levels to concrete security mechanisms and technical standards that are supported by the given platform. For example, a platform operator may map the security goal "Authentication" at the security requirement level "low" to a formal WS-SecurityPolicy statement which specifies that a "UsernameToken" with a "password digest" and a "creation time stamp" is required to be authenticated. The same security goal with the security requirement level "high" could, e.g., be mapped to a WS-SecurityPolicy statement demanding an "X.509-Token."

The security mechanisms as well as the mapping of security goals and requirement levels to security mechanisms and technical standards are likely to vary from platform to platform, depending on the application domain and on the technical security services and standards supported by the platform. For this reason, USDL does not prescribe the form or structure of a platform-specific security profile. It is simply referenced.

In summary, the specification of security requirement levels for security goals enables service providers to express business service security characteristics on a an abstract rather than on a technical level. The same applies to service consumers that may search for appropriate business services based on these abstract characteristics.

The following short example further refines the 2PL Airline Manager (cf. Example 8 in Section 8.7), and illustrates the utilization of security attributes. The "2PL Airline Manager" provides two interfaces for *looking up rates* and *kicking-off shipments*. It was decided, that the operation "looking up rates" should be public, and therefore does not require any security characteristics; resulting in the USDL specification as outlined in Listing 12.2 on page 326: lines 28 to 48. Whereas the kick-off shipment requires medium Authentication and high Encryption, as it has to be known, who initiated the shipment, and the data of the operation should be kept confidential.

The service provider of the 2PL Airline Manager utilizes the "platform-specific security profile," defined by the platform operator of the logistics marketplace. At first the service provider has to select the "platform-specific security profile," which is in this case an ontology.[7] Then, the service provider defines the desired security requirements by associating the adequate security goals with the correspondent

[7] Accessible via `http://ontology.logistics_service.org/security/lso_profile123`

security requirement levels (see Figure 12.6). The USDL serialization is also illustrated in Listing 12.2: lines 50 to 73.

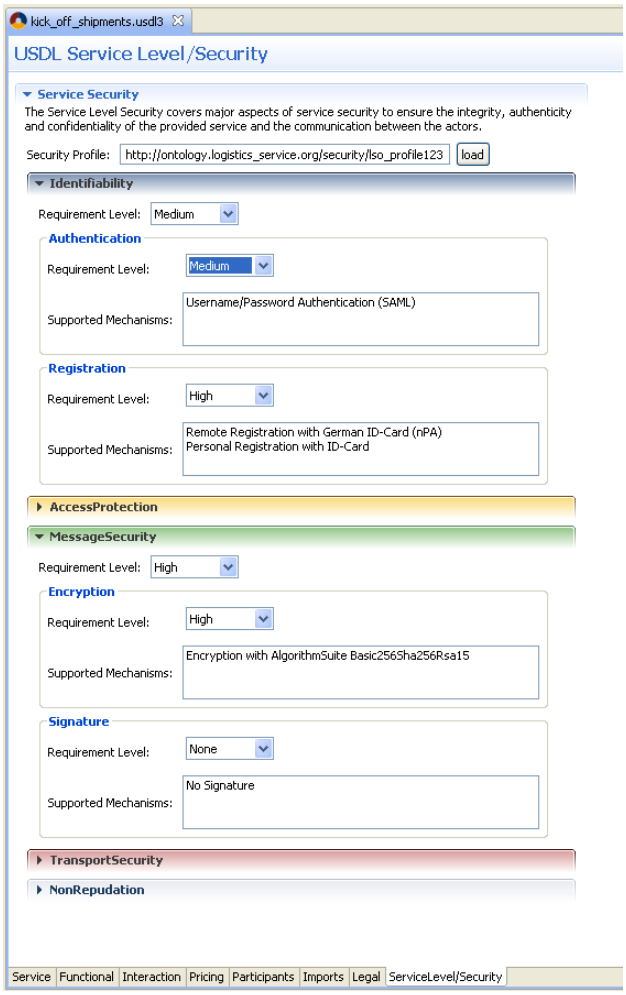

Fig. 12.6: Specifying security via SecurityAttributes with the USDL Editor (cf. Section 15.2.1).

12.4.2 SecurityMetric

Similar to the SecurityAttribute element, the SecurityMetric element specifies both a SecurityGoal and a RealizationLevel, but does not feature a SecurityRequirementLevel. Note that this is an alternative way to describe security requirements/claims, not intended to be used in combination with SecurityAttribute. Here, we do not need to express the requirement level, from low to high as in this approach we define more concrete security properties with link to technical artifacts. Security is then expressed in terms of concrete actions rather than an abstract level. Unlike SecurityAttribute, which only defines an abstract requirement level, the security metric enables a direct mapping with technical artifacts. A service provider can then specify some claims in terms of mechanism, such as internal procedure to erase Personally Identifiable Information after a certain amount of time to cover a privacy SecurityGoal. Also, the specifications to communicate with the service provider are no longer platform-specific, but rather described and decided when one operates service elicitation.

Having security described with technical artifacts allows actors that understand security protocols and standards to express their security demands in such terms. For example, during the matchmaking phase, a service consumer might restrict search to services that support data confidentiality at the application level through usage of a specific encryption algorithm. Prior to publishing the service, the provider defines which algorithms are accepted, such as AES and RSA for symmetric and asymmetric cryptography, and puts these capabilities in USDL. The consumer is then able to check which service is compatible with his requirements.

SecurityMetric defines what the service provider claims or requires in terms of technical artifacts. As this element is mostly for automated services, we can foresee usage of this element for manual or semi-automated services, such as making sure in a parcel shipment service that the warehouse clerk sets the seal on the box and then signs the registry.

Listing 12.3 on page 327 is an instantiation of the model shown in Figure 12.5. It provides information for an automated service on how identity is managed to certify authentication and provide message encryption to avoid leakage of data. The example is based on the example of the 2PL Airline Manager (cf. Section 8.7) where we try to specify more concrete mechanisms than previous section.

Instead of declaring the security level in an abstract way, we observe that the snippet gives us details about security mechanisms and goals. It automatically links security requirements with two external security policies, as outlined in Listing 12.3: lines 2 to 26. The first one is set to express encryption of a SOAP Body and the second adds support for SAML token for identity propagation, i.e., authentication. From line 35 to 38, we specify a protocol that is not an implementation specification. In our case, we introduce the HTTP Authentication scheme to express future usage of BASIC or DIGEST authentication. Then, lines 39 to 62 is the first block that expresses a concrete security mechanism. With the goal to describe authentication security at the session level, the block lists security mechanisms accepted by the service provider. Technologies can be linked coming from various sources, such as

the reference to SAML Token coming from a WS-SecurityPolicy or from protocol documentation such as the HTTP Auth link. USDL provides expression logic to decide and restrict application of security mechanisms. The second block in lines 63 to 75 indicates the usage of message confidentiality and contains a summary of what is used. The service consumer is able to quickly understand the profile used by the service, and in case of further and detailed information he can consult directly the referenced security policy.

12.5 Trustworthiness of Service Providers

In contrast to security measures, which are targeted at third party threats, trust considerations are concerned with the risk emanating from business partners, in particular, unknown service providers. Trust calculation in this field evaluates which of the suitable providers is most likely to actually deliver what was promised [7]. This definition of trust is in line with Marsh's dissertation thesis [15], viz., the earliest transfer of the concept from humanities to computer sciences. He defines trust as the confidence towards a decision that entails obvious risks.

In an Internet of Services environment, where automatic, manual, and hybrid services are traded, there are three conceivable sources that can be considered to estimate the amount of a trust a consumer should put into a particular offering. The actual calculation of a trust score depends highly on the domain and the consumers preferences, but in any domain and for any user, the following categories apply.

The first source is provider and service information supplied by the provider himself. This is precisely the data covered in a USDL service description. It can safely be called *objective*, since any piece of information can, at least in theory, be verified by a neutral third party. This type of source is dealt with in the following section.

As opposed to the objective data of the service description, trust can be based on subjective data, i.e., feedback of other users. The connection of USDL with this reputation-based trust is discussed in the subsequent Section 12.5.2.

The third possibility for service seekers to judge trustworthiness is run time data, i.e., the information that the service platform collects about providers, offerings and invocations as they are delivered over time. This material, however, is out of scope of the static service description which is the purpose of USDL.

12.5.1 Trust Directly Based in Service Description

The USDL service description may offer various cues for confidence in a service offering. In Chapter 13, the class Certification is introduced (see also Figure 12.7). By asking neutral third parties to issue certificates about agents or a resources a provider

can establish trustworthiness. Straightforward examples for such are TrustedShops[8] and organic food.[9] Ratings issued by agencies can also be modeled using the Certification class, such as Skytrax's World Airline Star Rating[10] or the stars of the European Hotelstars Union.[11]

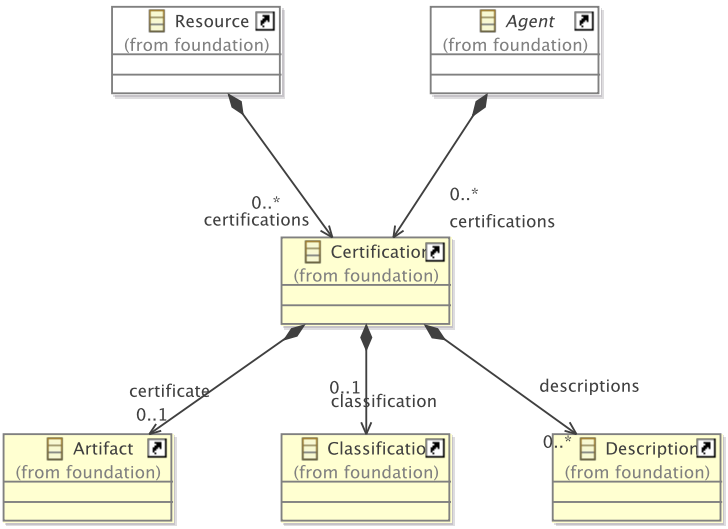

Fig. 12.7: Class Certification.

Inside the framework of USDL, there is no way to ensure that only rightful certifications are claimed. Nevertheless, it is reasonable to trust the claim of the certification as much as one trusts the neutral party that supposedly issued it: to make sure that no one falsely shows certificates is of vital business interest to the respective rating agencies and USDL makes it easy for them to scan for abuses. For example, TrustedShops can easily check if everyone referencing them actually shows in their files. Since we are dealing with an electronic market place and formalized service descriptions, this is much easier than in the case of physical shops that hang up physical documents.

Apart from these explicit trust cues, there is a range of USDL elements that implicitly induce some amount of trustworthiness. An example for this is the physical location of a provider. In the logistics domain, a courier company looking for an ocean carrier may not strictly require that it be based in a certain country, but it may find EU based carriers more trustworthy than American competitors. Likewise the

[8] http://www.trustedshops.de

[9] http://www.bio-siegel.de

[10] http://www.airlinequality.com/StarRanking/ranking.htm

[11] http://www.hotelstars.eu

attribute yearOfFounding in class Organization can contribute to the trust calculation. Beyond these there are various pieces of information contained in the USDL description, that can be exploited for trust calculation. However, that calculation depends highly on the domain and on consumer preferences and is therefore out of the scope of this book. In the THESEUS/TEXO project, we realized a sample trust prediction for a car rental scenario [10] based among on USDL data. Similar to certification, all this information is provider supplied and hence questionable. So reasonable trust preferences put more weight on such certain elements; namely those that are likely to be audited by a relevant actor, who has a natural incentive. The location, for example, is most likely to be checked by the service market place operator.

On a conceptual level, a clear distinction can be made between two classes of USDL items: on the one hand, items that imply how well an offering matches the demand of a service seeker. On the other hand, items that assures confidence in the provider. An example of the former is the price, the latter could be a certificate. In practice, however, the line is blurry, certain elements can easily appeal to both the liking and the trusting of a service seeker. Imagine a traveler looking for a hotel: A two-star cuisine may not be a requirement, yet its presence can be interpreted as a sign of overall reliability. In other circumstances one piece of information can even contribute in a contradictory way to preference and reliability. To illustrate this, let us consider a 3PL courier company (cf. running example from Chapter 8) that seeks an airline service. Given their functional parameters such as origin, destination, dates, weight, etc. there might be ten airlines offering that particular service at a particular price. Now, while a lower price is quite to the liking of the seeker, a fee lower than half the mean price may be a cue to distrust that provider.

12.5.2 Using the Service Description to Harness User Ratings

A different approach to evaluate the trustworthiness of a provider is to consider his reputation, i.e., the reported experience of other users. This field is currently still under investigation and a recent overview about reputation in service-oriented environments is provided in [13]. The essential idea is that of wisdom of crowds [22]: the more users report on a given provider, the better his reputation predicts his behavior. Additionally, and in contrast to the static data mentioned in the previous section, a reputation system makes white-washing difficult, i.e., the cost of building up a trusted profile can stop malicious providers from cheating with fresh accounts [4].

The actual reputation can obviously not be part of a static USDL service description, since it is highly dynamic. Nevertheless, USDL can be used to remedy some of the problems often encountered in reputation systems.

In many domains a simple measure of how happy a consumer was with a service is too undifferentiated. This also applies if the rating system is used for other purposes than trust calculation. For instance, providers want specific aspects of their services rated to guide their innovation process. As these two were the main applica-

tions for feedback information foreseen by the USDL meta-modelers, the elements discussed here cater for the needs of both. Consequently, a framework for evaluating specific aspects of a service or provider is needed. Moreover, domain specific scales may be necessary, wherever a simple five star scale is insufficient. On the other hand, there is a mechanism that ensures that comparable services can be rated in a consistent way. Otherwise, providers would make their offering ratable only in those categories where they excel.

In order to meet the requirement of *Extensibility* (cf. Section 8.3.1.4) a complete feedback meta-model was developed in THESEUS/TEXO [14], that allows for the definition of a hierarchy of rating aspects on a corresponding scale. Since that meta-model is not specific to the realm of services, but could also be used for products, it was not incorporated in USDL. Instead, instances of the feedback meta-model can be referenced using the USDL class Artifact.

The interlinking of feedback with relevant USDL classes is depicted in Figure 12.8. Essentially, the relation feedbackModels adds 0 to *n* feedback model Artifacts to NetworkProvisionedEntity, AbstractService and Agent. ServiceBundle and Service inherit the link from NetworkProvisionedEntity and they represent the core objects to be rated. By means of the super class Agent, USDL users can describe how Persons and Organizations can be rated. The class AbstractService deserves a more detailed discussion.

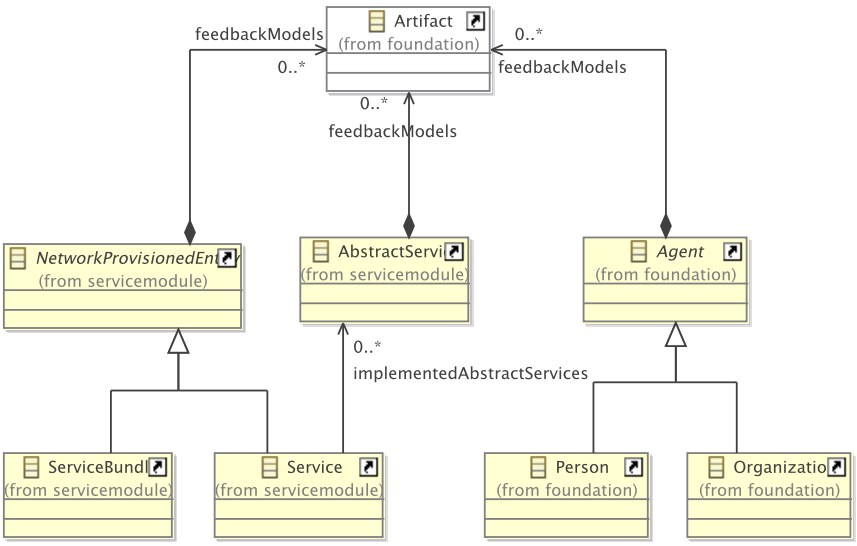

Fig. 12.8: Model elements introduced for a feedback infrastructure.

The feedback models attached to AbstractService enable consist rating schemes across similar services across multiple providers. A domain knowledgeable author-

ity such as the service market place platform operator can define AbstractServices and specify how instances of these are to be evaluated. Concrete services claim to implement a given AbstractService in order to be suggested to seekers of this kind of offering. By that they automatically are subject to the rating scheme attached to the abstract one, and hence all competing services are guaranteed to sharing the most important evaluation aspects for this kind of offering. Moreover, providers can attach further rating models inside their USDL description to get a more detailed feedback on their performance.

To illustrate this, let us consider an example. We chose a domain where most principals are unknown to each other, and hence must base their trust on other consumers' rating. This diverges from the running example where principals are most likely to have some past interaction and do not have to rely on third party opinions. Consider a service market that trades automotive services. The platform host would reasonably create an AbstractService "car repair" and link it up with a feedback model that covers at least "quality of repair" and "speed of repair." Thus, all providers that want to be taken into account when a car repair is wanted, must reference this AbstractService and can therefore automatically be evaluated in these two most relevant categories. This in turn leads to a consistent reputation landscape of car repair services, which is suitable for service seekers to base their trust on.

12.6 Conclusion

In this chapter we presented an overview of existing approaches to model service levels with an extra glimpse on security specific languages. Building on that, we described how USDL addresses existing gaps, followed by a discussion of how the Service Level Module in constructed and how it relates to other USDL elements. Again, special care was taken to discuss how security properties are covered in USDL. Namely, USDL allows service providers to specify security offerings and state their security requirements.

Subsequently, we explained the USDL meta-model elements that are related to security and trust in detail. Most notably the classes SecurityAttribute and SecurityMetric for specifying security goals either on a high level or on a technical level, respectively. The relation feedbackModel was introduced explicitly to facilitate trust calculation. It associates services with rating schemes. Additionally, we surveyed preexisting USDL elements that offer cues for trust estimation.

We decided that trust and security elements do not form a USDL module of their own, since they do not represent a group of functional elements. They rather form part of the Service Level Module which host most non-functional properties.

Based on the foundations laid out in this chapter, we see several lines of future work: First, the discussed security properties are on a merely technical level and are not well suited for describing security properties on a more abstract, i.e., business level. On a business level, users usually do not want to specify properties such as Confidentiality or Authorization. Instead, they want to specify more abstract prop-

erties such as "comply to the following regulations" or "it is only allowed to share this data between the following parties." Supporting such high-level specifications requires, on the one hand, to link the security extension with the Legal Module (see Chapter 10). This allows for an extensive support of legal compliance regulations such as the Sarbanes-Oxley Act [20] in the financial industry or HIPAA [6] in the health care industry. Supporting such legal compliance regulations is particular challenging as they combine legal requirements based on abstract concepts with technical security and privacy aspects. On the other hand, this requires a process for mapping high-level requirements to technical realizations of those requirements and, thus, allow business experts and security experts to work together for providing secure, trustworthy, and compliant applications on top of the Internet of Services. Second, we plan to provide extensions that serve domain specific needs, e.g., that allow for describing advanced access control or privacy needs in the health care domain. The reader interested in a detailed description of such requirements is, e.g., referred to the documents describing the security requirements England's National Programme for Information Technology (NPfIT) of the National Health Service (NHS) [18, 19]. Third, we plan to empirically evaluate the concepts introduced here, i.e., to investigate how well they help to describe and trade business services.

References

1. A. Andrieux, K. Czajkowski, A. Dan, et al. Web services agreement specification (ws-agreement). Technical report, OpenGridForum, 2007. `http://www.ogf.org/documents/GFD.107.pdf`.
2. M. G. Buscemi and U. Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In *Programming Languages and Systems, 16th European Symposium on Programming*, pages 18–32. Springer, 2007.
3. J. Day and H. Zimmermann. The OSI Reference Model. *Proceedings of the IEEE*, 71(12):1334–1340, Dec. 1983.
4. M. Feldman and J. Chuang. The Evolution of Cooperation under Cheap Pseudonyms. In *7th IEEE International Conference on E-Commerce Technology (CEC)*, pages 284–291, München, jul 2005. IEEE Computer Society.
5. D. Gregg and S. Walczak. The relationship between website quality, trust and price premiums at online auctions. *Electronic Commerce Research*, 10:1–25, 2010.
6. HIPAA. Health Insurance Portability and Accountability Act of 1996. `http://www.cms.hhs.gov/HIPAAGenInfo/`, 1996.
7. A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.
8. K. Kearney, F. Torelli, and C. Kotsokalis. SLA*: An Abstract Syntax for Service Level Agreements. In *Proceedings of IEEE Grid2010 conference; Service Level Agreements in Grids Workshop*, Brussels, 2010.
9. D. D. Lamanna, J. Skene, and W. Emmerich. Slang: A language for defining service level agreements. In *Future Trends in Distributed Computing Systems*, pages 100–106. IEEE Computer Society, 2003.
10. E. Lapi, E. Höfig, and F. Marienfeld. THESEUS/TEXO Consortium: TRICE Deliverable E4 – Demonstrator Based on TEXO Platform, Feb. 2011.

11. K. Lawrence, C. Kaler, and al. Ws-securitypolicy 1.3. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.html, 2009.
12. H. Ludwig, A. Keller, A. Dan, et al. Web service level agreement (wsla) language specification. Technical report, IBM Research, 2003. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.
13. Z. Malik and A. Bouguettaya. *Trust Management for Service-Oriented Environments*. Springer US, 2009.
14. F. Marienfeld, E. Höfig, and E. Lapi. THESEUS/TEXO Consortium: TRICE Deliverable E2 – Extension of USDL with Trust and Quality Criteria, Feb. 2011.
15. S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD Thesis, University of Stirling, 1994.
16. T. Moses. eXtensible Access Control Markup Language(XACML) Version 2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdfl, 2005.
17. L. Nixon, D. Lambert, A. Filipowska, and E. Simperl. Future of the Internet of Services for Industry: the ServiceWeb 3.0 Roadmap. *Future Internet Assembly (FIA 2009)*, 2009.
18. D. of Health. The Care Record Guarantee. Our Guarantee for NHS Care Records in England. Technical report, Department of Health, 2009.
19. D. of Health. Information Governance (IG) Concepts, 2010. http://www.connectingforhealth.nhs.uk/systemsandservices/infogov/.
20. P. Sarbanes, G. Oxley, et al. Sarbanes-Oxley Act of 2002. 107th Congress Report, House of Representatives, 2nd Session, 107–610, 2002.
21. F. I. P. SLA@SOI. Empowering the service industry with sla-aware infrastructures. http://sla-at-soi.eu.
22. J. Surowiecki. *The Wisdom of Crowds*. Doubleday, 2004.
23. the STORK-eid Consortium. STORK Deliverable D2.1 - Framework Mapping of Technical/Organisational Issues to a Quality Scheme. https://www.eid-stork.eu/index.php?option=com_processes&Itemid=&act=streamDocument&did=579, 2011.
24. S. Venugopal, X. Chu, and R. Buyya. A Negotiation Mechanism for Advance Resource Reservations Using the Alternate Offers Protocol. In *16th International Workshop on Quality of Service*, pages 40–49, june 2008.

Listings

Listing 12.1: USDL Service Level Agreement sample

```
<identifiableElement xsi:type="service:Service">

  <contextVariables>
    <!-- variable for expected duration of delivery in weeks,
         default: 1 week -->
    <variableDeclaration xsi:id="varExpDelDuration">
      <name>
        <value> expectedDeliveryDuration </value>
        <type> name </type>
      </name>
      <defaultValue> 1 </defaultValue>
      <typeReference>
        <classificationSystemID> http://www.internet-of-services.com/
  serviceTypes </classificationSystemID>
        <classID> duration_week </classID>
        <unitSymbol> wk </unitSymbol>
        <descriptions>
```

```
18          <description>
19            <value> duration in weeks </value>
20            <type> freetextShort </type>
21            <language> en </language>
22          </description>
23        </descriptions>
24      </typeReference>
25    </variableDeclaration>
26  </contextVariables>
27
28  <serviceLevelProfiles>
29    <serviceLevelProfile>
30
31      <serviceLevels>
32
33        <!-- service level #1 -->
34        <serviceLevel xsi:type="servicelevel:GuaranteedState">
35          <!-- provider is obligated -->
36          <obligatedParty> prov321 </obligatedParty>
37
38          <descriptions>
39            <description>
40              <value> Delivery duration must not longer than what has been
41                      specified by the customer. </value>
42              <type> freetextShort </type>
43              <language> en </language>
44            </description>
45          </descriptions>
46
47          <stateSpecification>
48
49            <!-- measured delivery duration is less than or equal to what is
50                 set in the service context (cust. input -->
51            <value> metric["m_delDuration"] <= variable["v_expDelDur"] </value>
52            <languageID> urn:example:expression_language </languageID>
53
54            <attributes>
55              <!-- reference to the variable -->
56              <attribute xsi:id="v_expDelDur"
57                         xsi:type="servicelevel:VariableReference">
58                <reference> varExpDelDuration </reference>
59              </attribute>
60              <!-- metric for measuring delivery duration -->
61              <attribute xsi:id="m_delDuration" xsi:type="servicelevel:Metric">
62                <typeReference>
63                  <classificationSystemID>
64    http://www.internet-of-services.com/serviceTypes
65                  </classificationSystemID>
66                  <classID> duration_day </classID>
67                  <unitSymbol> d </unitSymbol>
68                  <descriptions>
69                    <description>
70                      <value> duration in days </value>
71                      <type> freetextShort </type>
72                      <language> en </language>
73                    </description>
74                  </descriptions>
75                </typeReference>
76                <assessment>
77                  <value> delivery duration as measured by receiving party
78                  </value>
79                  <type> freetextLong </type>
80                  <language> en </language>
81                </assessment>
82              </attribute>
83            </attributes>
84
```

```
85          </stateSpecification>
86        </serviceLevel>
87
88        <!-- service level #2 -->
89        <serviceLevel xsi:type="servicelevel:GuaranteedState">
90          <!-- provider is obligated -->
91          <obligatedParty> prov321 </obligatedParty>
92
93          <descriptions>
94            <description>
95              <value> The temperature of the goods is maintained at minus 5
96                      degrees. </value>
97              <type> freetextShort </type>
98              <language> en </language>
99            </description>
100         </descriptions>
101
102         <stateSpecification>
103
104           <!-- measured temperature is approximately equal to minus 5
105                degrees Celsius -->
106           <value> metric["m_temp"] ~= constant["c5"] </value>
107           <languageID> urn:example:expression_language </languageID>
108
109           <attributes>
110             <!-- temperature threshold (modeled as constant) -->
111             <attribute xsi:id="c5" xsi:type="servicelevel:Constant">
112               <value> -5 </value>
113               <typeReference>
114                 <classificationSystemID>
115                   http://www.internet-of-services.com/serviceTypes
116                 </classificationSystemID>
117                 <classID> temperature_celsius </classID>
118                 <unitSymbol> dC </unitSymbol>
119                 <descriptions>
120                   <description>
121                     <value> temperature in degrees Celsius </value>
122                     <type> freetextShort </type>
123                     <language> en </language>
124                   </description>
125                 </descriptions>
126               </typeReference>
127             </attribute>
128             <!-- metric for measuring temperature, related to goods (input to
129                  service function "Transport") -->
130             <attribute xsi:id="m_temp" xsi:type="servicelevel:Metric">
131               <relatesTo> paramGoods </relatesTo>
132               <typeReference>
133                 <classificationSystemID>
134                   http://www.internet-of-services.com/serviceTypes
135                 </classificationSystemID>
136                 <classID> temperature_celsius </classID>
137                 <unitSymbol> dC </unitSymbol>
138                 <descriptions>
139                   <description>
140                     <value> temperature in degrees Celsius </value>
141                     <type> freetextShort </type>
142                     <language> en </language>
143                   </description>
144                 </descriptions>
145               </typeReference>
146               <assessment>
147                 <value> temperature of goods as constantly measured
148                         by provider </value>
149                 <type> freetextLong </type>
150                 <language> en </language>
151               </assessment>
```

```
152                </attribute>
153              </attributes>
154
155            </stateSpecification>
156          </serviceLevel>
157
158          <!-- service level #3 -->
159          <serviceLevel xsi:type="servicelevel:GuaranteedAction">
160            <!-- provider is obligated -->
161            <obligatedParty> prov321 </obligatedParty>
162
163            <descriptions>
164              <description>
165                <value> Penalty payment of EUR100 per full day of delayed
                       delivery
166                </value>
167                <type> freetextShort </type>
168                <language> en </language>
169              </description>
170            </descriptions>
171
172            <preconditionSpecification>
173              <!-- measured delivery delay is greater or equal than 1 -->
174              <value> metric["m_delDelay"] >= constant["c6"] </value>
175              <languageID> urn:example:expression_language </languageID>
176
177              <attributes>
178                <!-- delivery delay threshold (modeled as constant) -->
179                <attribute xsi:id="c6" xsi:type="servicelevel:Constant">
180                  <value> 1 </value>
181                  <typeReference> <!-- same type description as specified in
182                         metric below --> </typeReference>
183                </attribute>
184                <!-- metric for measuring delay in delivery  -->
185                <attribute xsi:id="m_delDelay" xsi:type="servicelevel:Metric">
186                  <typeReference>
187                    <classificationSystemID>
188                         http://www.internet-of-services.com/serviceTypes
189                    </classificationSystemID>
190                    <classID> duration_day </classID>
191                    <unitSymbol> d </unitSymbol>
192                    <descriptions>
193                      <description>
194                        <value> duration in days </value>
195                        <type> freetextShort </type>
196                        <language> en </language>
197                      </description>
198                    </descriptions>
199                  </typeReference>
200                  <assessment>
201                    <value>delayed delivery as measured by receiving party</value
                         >
202                    <type> freetextLong </type>
203                    <language> en </language>
204                  </assessment>
205                </attribute>
206              </attributes>
207            <preconditionSpecification>
208
209            <actionSpecification>
210              <value> <!-- credit the customer with (EUR 100
                     * floor(metric["m_delDelay"])) --> </value>
211              <languageID> urn:example:action_language </languageID>
212            </actionSpecification>
213
214          </serviceLevel>
215        </serviceLevels>
216
```

```
    </serviceLevelProfile>
  </serviceLevelProfiles>


</identifiableElement>
```

Listing 12.2: USDL SecurityAttribute

```
<!-- ... -->
<serviceLevelProfile>
  <implementationSpecifications>
    <implementationSpecification xsi:id="lso_profile123">
      <type> TechnicalMetadata </type>
      <mimeType> application/xml </mimeType>
      <uri> http://ontology.logistics_service.org/security/lso_profile123 </uri
          >
      <descriptions>
        <description>
          <value> security profile 123 </value>
          <type> name </type>
          <language> en </language>
        </description>
      </descriptions>
    </implementationSpecification>
  </implementationSpecifications>
<!-- ... -->
  <serviceLevels>
    <serviceLevel xsi:type="servicelevel:GuaranteedState">
       <!-- ... -->
      <stateSpecification>
        <!-- ... -->
        <value> attribute["sec1"] applies </value>
        <value> attribute["sec2"] applies </value>
         <!-- ... -->
        <attributes>
         <!-- look_up_rates has no security restrictions -->
          <attribute xsi:id="sec1" xsi:type="slbaseext:SecurityAttribute">
            <relatesTo>airline_mgmt:look_up_rates</relatesTo>
            <securityGoals>
              <securityGoal>Authentication</securityGoal>
            </securityGoals>
            <requirementLevel>none</requirementLevel>
            <platformSpecificSecurityProfile>
              lso_profile123
            </platformSpecificSecurityProfile>
          </attribute>
          <attribute xsi:id="sec2" xsi:type="slbaseext:SecurityAttribute">
            <relatesTo>airline_mgmt:look_up_rates</relatesTo>
            <securityGoals>
              <securityGoal> Confidentiality </securityGoal>
            </securityGoals>
            <requirementLevel>none</requirementLevel>
            <platformSpecificSecurityProfile>
              lso_profile123
            </platformSpecificSecurityProfile>
          </attribute>
          <!-- ... -->
          <!-- kick_off_shipments requires medium Authentication
               and high Confidentiality
           -->
          <attribute xsi:id="sec3" xsi:type="slbaseext:SecurityAttribute">
            <relatesTo>airline_mgmt:kick_off_shipments</relatesTo>
            <securityGoals>
              <securityGoal> Authentication </securityGoal>
            </securityGoals>
```

```
58              <requirementLevel>medium</requirementLevel>
59              <platformSpecificSecurityProfile>
60                lso_profile123
61              </platformSpecificSecurityProfile>
62            </attribute>
63            <attribute xsi:id="sec4" xsi:type="slbaseext:SecurityAttribute">
64              <relatesTo>airline_mgmt:kick_off_shipments</relatesTo>
65              <securityGoals>
66                <securityGoal>Confidentiality</securityGoal>
67              </securityGoals>
68              <requirementLevel>high</requirementLevel>
69              <platformSpecificSecurityProfile>
70                lso_profile123
71              </platformSpecificSecurityProfile>
72            </attribute>
73          </attributes>
74          <!-- ... -->
75        </stateSpecification>
76      </serviceLevel>
77    </serviceLevels>
78  </serviceLevelProfile>
79  <!-- ... -->
```

Listing 12.3: USDL Security Metric

```
1   <serviceLevelProfile>
2     <implementationSpecifications>
3       <implementationSpecification xsi:id="lso_Wssp1.2_EncryptBody">
4         <type> TechnicalMetadata </type>
5         <mimeType> application/xml </mimeType>
6         <uri> http://logistics_service.org/security/lso-Wssp1.2-EncryptBody.xml
                </uri>
7         <descriptions>
8           <description>
9             <value> WS-SecurityPolicy to express that the entire body of a soap
10            message has to be encrypted </value>
11            <language> en </language>
12          </description>
13        </descriptions>
14      </implementationSpecification>
15      <implementationSpecification xsi:id="lso_Wssp1.2_SupportSAMLToken">
16        <type> TechnicalMetadata </type>
17        <mimeType> application/xml </mimeType>
18        <uri> http://logistics_service.org/security/lso-Wssp1.2-SupportSAMLToken.
                xml </uri>
19        <descriptions>
20          <description>
21            <value> WS-SecurityPolicy to express the support of SAML token for
                    Identity </value>
22            <language> en </language>
23          </description>
24        </descriptions>
25      </implementationSpecification>
26    </implementationSpecifications>
27    <!-- -->
28    <serviceLevels xsi:type="servicelevel:GuaranteedState" obligatedParty="//
          @Roles.0">
29      <stateSpecification>
30        <!-- ... -->
31        <attributes>
32          <!-- kick_off_shipments requires medium and
33                 high Confidentiality
34               -->
35          <attribute xsi:id="sec:httpauth" xsi:type="servicelevel:GenericConstant
                ">
36            <!-- HTTP AUTH (BASIC or DIGEST) -->
37            <value>http://www.ietf.org/rfc/rfc2617.txt</value>
```

```
38          </attribute>
39          <attribute xsi:id="sec5" xsi:type="slbaseext:SecurityMetric">
40            <relatesTo>airline_mgmt:kick_off_shipments</relatesTo>
41            <securityGoals>
42              <securityGoal> Authentication </securityGoal>
43            </securityGoals>
44            <RealizationLevel> Session </RealizationLevel>
45            <securityMechanisms>
46              <securityMechanism xsi:type="sec:httpauth">
47                <value>HTTP AUTH</value>
48              </securityMechanism>
49              <securityMechanism xsi:type="lso_Wssp1.2_SupportSAMLToken">
50                <value>SAML</value>
51              </securityMechanism>
52            </securityMechanisms>
53            <expressionSpecification>
54              <description>
55              Authentication is verified through HTTP  headers
56              or token in SOAP messages
57            </description>
58              <expression>
59              sec:httpauth OR lso_Wssp1.2_SupportSAMLToken
60            </expression>
61              </expressionSpecification>
62          </attribute>
63          <attribute xsi:id="sec6" xsi:type="slbaseext:SecurityMetric">
64            <relatesTo>airline_mgmt:kick_off_shipments</relatesTo>
65            <securityGoals>
66              <securityGoal>Confidentiality</securityGoal>
67            </securityGoals>
68            <RealizationLevel> Message </RealizationLevel>
69            <securityMechanisms>
70              <securityMechanism xsi:type="lso_Wssp1.2_EncryptBody">
71                <type>Encryption</type>
72                <value>Basic256Sha256Rsa15</value>
73              </securityMechanism>
74            </securityMechanisms>
75          </attribute>
76        </attributes>
77        <!-- ... -->
78      </stateSpecification>
79      <!-- ... -->
80    </serviceLevels>
81  </serviceLevelProfile>
```